

Dealing with bad apples: Robust range-based network localization via distributed relaxation methods

Cláudia Soares*, *Member, IEEE*, and João Gomes, *Member, IEEE*

Abstract—Real-world network applications must cope with failing nodes, malicious attacks, or, somehow, nodes facing corrupted data — classified as outliers. One enabling application is the geographic localization of the network nodes. However, despite excellent work on the network localization problem, prior research seldom considered outlier data — even now, when already deployed networks cry out for robust procedures. We propose robust, fast, and distributed network localization algorithms, resilient to high-power noise, but also precise under regular Gaussian noise. We use the Huber M-estimator as a difference measure between the distance of estimated nodes and noisy range measurements, thus obtaining a robust (but nonconvex) optimization problem. We then devise a convex underestimator solvable in polynomial time, and tight in the inter-node terms. We also provide an optimality bound for the convex underestimator. We put forward a new representation of the Huber function composed with a norm, enabling distributed robust localization algorithms to minimize the proposed underestimator. The synchronous distributed method has optimal convergence rate and the asynchronous one converges in finite time, for a given precision. The main highlight of our contribution lies on the fact that we pay no price for distributed computation nor in accuracy, nor in communication cost or convergence speed. Simulations show the advantage of using our proposed algorithms, both in the presence of outliers and under regular Gaussian noise: our method exceeds the accuracy of an alternative robust approach based on L1 norms by at least 100m in an area of 1Km sides.

Index Terms—Distributed algorithms, Robust estimation, Huber function, convex relaxations, nonconvex optimization, maximum likelihood estimation, distributed iterative network localization, sensor networks.

EDICS Category: OPT-CVXR OPT-DOPT NEG-APPL
NEG-LOCL

I. INTRODUCTION

Outliers can cause large errors in non robust estimation algorithms, and, if other systems use wrong estimates as input, error propagation can invalidate the the engineered system's final purpose. Network localization is a key component in many network-centric systems that is prone to such catastrophic error propagation. It might be taken for granted in most sensor network applications, but in challenging environments network localization is an open and very active research field. We present a new approach addressing the presence of outliers, not by eliminating them from the estimation process, but by

weighting them, so they can contribute to the solution, while mitigating the outlier bias on the estimator.

A. The problem

The network is represented as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We represent the set of sensors with unknown positions as $\mathcal{V} = \{1, 2, \dots, n\}$. There is an edge $i \sim j \in \mathcal{E}$ between nodes i and j if a range measurement between i and j is available and i and j can communicate with each other. Anchors have known positions and are collected in the set $\mathcal{A} = \{1, \dots, m\}$; they are not nodes on the graph \mathcal{G} . For each sensor $i \in \mathcal{V}$, we let $\mathcal{A}_i \subset \mathcal{A}$ be the subset of anchors with measured range to node i . The set N_i collects the neighbor sensor nodes of node i .

The element positions belong to \mathbb{R}^p with $p = 2$ for planar networks, and $p = 3$ for volumetric ones. We denote by $x_i \in \mathbb{R}^p$ the position of sensor i , and by d_{ij} the range measurement between sensors i and j . Anchor positions are denoted by $a_k \in \mathbb{R}^p$. We let r_{ik} denote the noisy range measurement between sensor i and anchor k .

We aim at estimating the sensor positions $x = \{x_{\mathcal{V}}\}$, taking into account two types of noise: (1) regular Gaussian noise, and (2) outlier induced noise.

B. Related work

Focusing on recent work, several different approaches are available, some performing semi-definite or weaker second-order cone relaxations of the original nonconvex problem like Oğuz-Ekim *et al.* [1] or Biswas *et al.* [2]. These approaches do not scale well, since the centralized semidefinite program (SDP) or second-order cone program (SOCP) gets very large even for a small number of nodes. In Oğuz-Ekim *et al.* the Majorization-Minimization (MM) framework was used with quadratic cost functions to also derive centralized approaches to the sensor network localization problem. Other approaches rely on multidimensional scaling, where the sensor network localization problem is posed as a least-squares problem, as in Shang *et al.* [3]. Unfortunately, multidimensional scaling is unreliable in large-scale networks with sparse connectivity. Also relying on the well-tested weighted least-squares approach, the work of Destino and Abreu [4] performs successive minimizations of a weighted least-squares cost function convolved with a Gaussian kernel of decreasing variance, following an homotopy scheme. Another class of relaxations are convex envelopes of terms in the cost function, like Soares *et al.* [5].

This research was partially supported by Fundação para a Ciência e a Tecnologia (project UID/EEA/50009/2013) and EU-H2020 project WiMUST (grant agreement No. 645141) The authors are with the Institute for Systems and Robotics (ISR), Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisboa, Portugal (e-mail: {csoares,jpg}@isr.tecnico.ulisboa.pt).

Some previous references directly tackle the nonconvex maximum likelihood problem, aspiring to no more than a local minimizer, whose goodness depends on the quality of the initialization. Here we can point out several approaches, like Costa *et al.* [6], where the authors present a distributed refinement solution inspired in multidimensional scaling, or Calafiore *et al.* [7], presenting a gradient algorithm with Barzilai-Borwein step sizes calculated in a first consensus phase at every algorithm step, and Soares *et al.* [8], where the authors reformulate the problem to obtain a Lipschitz gradient cost; shifting to this cost function enables a MM approach based on quadratic upper bounds that decouple across nodes; the resulting algorithm is distributed, with all nodes working in parallel.

All these approaches assume Gaussian noise contaminating the distance measurements or their squares, while many empirical studies reveal that real data are seldom Gaussian. Despite this, the literature is scarce in robust estimation techniques for network localization. Some of the few approaches rely on identifying outliers from regular data and discarding them. An example is Ihler *et al.* [9], which formulates network localization as an inference problem in a graphical model. To approximate an outlier process the authors add a high-variance Gaussian to the Gaussian mixtures and employ non-parametric belief propagation to approximate the solution. The authors assume a particular probability distribution for outlier measurements. In the same vein, Ash *et al.* [10] employs the EM algorithm to jointly estimate outliers and sensor positions. Recently, the work of Yin *et al.* [11] tackled robust localization with estimation of positions, mixture parameters, and outlier noise model for unknown propagation conditions, again under predetermined probability distributions. By removing guessed outliers from the estimation process some information is lost.

Alternatively, methods may perform a soft rejection of outliers, still allowing them to contribute to the solution. Oğuz-Ekim *et al.* [1] derived a maximum likelihood estimator for Laplacian noise and relaxed it to a convex program by linearizing and dropping a rank constraint; they also proposed a centralized algorithm to solve the approximated problem. Such centralized solutions fail to scale with the number of nodes and number of collected measurements. Forero and Giannakis [12] presented a robust multidimensional scaling based on regularized least-squares, where the regularization term was surrogated by a convex function, and solved via MM. The main drawbacks of this approach are the centralized processing architecture and selection of a sensitive regularization parameter. Korkmaz and van der Veen [13] use the Huber loss [14] composed with a discrepancy between measurements and estimated distances, in order to achieve robustness to outliers. The resulting cost is nonconvex, and optimized by means of the Majorization-Minimization technique. The method is distributed, but the quality of the solution depends in the quality of the initialization. Yousefi *et al.* [15] extends the Projection Onto Convex Sets approach in Blatt and Hero [16] to the Huber loss. The approximation is then solved via a coordinate descent algorithm, where a “one-at-a-time” (sequential) update scheme is critical for convergence; so this solution depends on knowledge of a Hamiltonian path

in the network — a known NP-complete problem.

C. Contributions

In applications of large-scale networks there is a need for a distributed localization method for soft rejection of outliers that is simple, scalable and efficient under any outlier noise distribution. The method we present incorporates outliers into the estimation process and does not assume any statistical outlier model. We capitalize on the robust estimation properties of the Huber function but, unlike Korkmaz and van der Veen, we do not address the nonconvex cost in our proposal, thus removing the initialization uncertainty. Instead, we derive a convex relaxation which numerically outperforms state-of-the-art methods, and other natural formulations of the problem. The contributions of this work are:

- 1) We motivate a tight convex underestimator for each term of the robust discrepancy measure for sensor network localization (Section III);
- 2) We provide an optimality bound for the convex relaxation, and we analyze the tightness of the convex approximation. We also compare it with other discrepancy measures and appropriate relaxations. All measurements contribute to the estimate, although we do not compute specific weights. Numerical simulations illustrate the quality of the convex underestimator (Section III-A);
- 3) We put forth a new representation of the Huber function composed with a norm (Section IV-A);
- 4) Capitalizing on the previous contributions, we develop a gradient method which is distributed, requires only simple computations at each node, and has guaranteed optimal convergence rate (Sections IV-D, and V-A);
- 5) Further, we introduce an asynchronous method for robust network localization, with convergence guarantees (Sections IV-E, and V-B);
- 6) We benchmark our algorithms with the state-of-the-art in robust network localization, achieving better performance with fewer communications (Section VI).

Both solutions proposed in this paper do not assume knowledge of a Hamiltonian path in the network. Also, the proposed scheme has the fastest possible convergence for a first order method, in the synchronous case, without degradation of accuracy. This is possible after analyzing the novel representation of the robust problem, as described in the sequel, and uncovering that the problem is, in fact, naturally distributed.

II. DISCREPANCY MEASURE

The maximum-likelihood estimator for sensor positions with additive i.i.d. Gaussian noise contaminating range measurements is the solution of the optimization problem

$$\underset{x}{\text{minimize}} \ g_G(x),$$

where

$$g_G(x) = \sum_{i \sim j} \frac{1}{2} (\|x_i - x_j\| - d_{ij})^2 + \sum_i \sum_{k \in \mathcal{A}_i} \frac{1}{2} (\|x_i - a_k\| - r_{ik})^2.$$

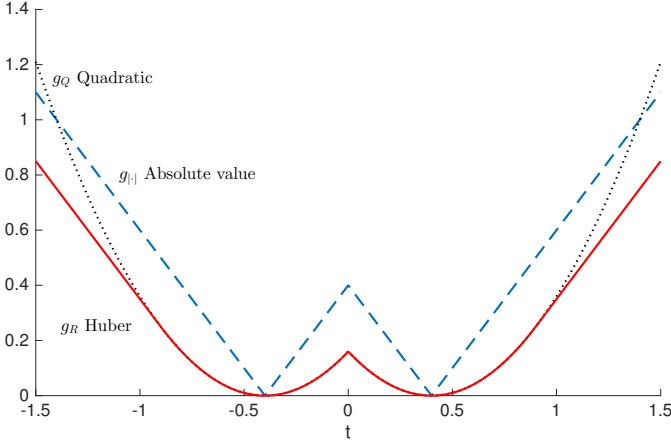


Fig. 1. The different cost functions considered in this paper, applied to a 1D, one-edge problem, where an anchor sits at the origin, and the sensor at 0.4. The maximum-likelihood independent white Gaussian noise is $g_Q(t) = (|t| - d)^2$ and shows the steepest tails, which act as outlier amplifiers; the L_1 loss $g_{|\cdot|}(t) = ||t| - d|$, associated with impulsive noise, fails to model the Gaussianity of regular operating noise; and, finally, the Huber loss $g_R(t) = h_R(|t| - d)$, combines robustness to high-power outliers and adaptation to medium-power Gaussian noise.

However, outlier measurements are non-Gaussian and will heavily bias the solutions of the optimization problem since their magnitude will be amplified by the squares $h_Q(t) = t^2$ in each outlier term. Robust estimation theory provides some alternatives to perform soft rejection of outliers, namely, using the L_1 loss $h_{|\cdot|}(t) = |t|$ or the Huber loss

$$h_R(t) = \begin{cases} t^2 & \text{if } |t| \leq R, \\ 2R|t| - R^2 & \text{if } |t| \geq R. \end{cases} \quad (1)$$

The Huber loss achieves the best of two worlds: it is robust for large values of the argument — like the L_1 loss — and for reasonable noise levels it behaves like g_Q , thus leading to the maximum-likelihood estimator adapted to regular noise. Figure 1 depicts a one-dimensional example of these different costs. We can observe in this simple example the main properties of the different cost functions, in terms of adaptation to low/medium-power Gaussian noise and high-power outlier spikes. Using (1) we can write our robust optimization problem as

$$\underset{x}{\text{minimize}} \ g_R(x) \quad (2)$$

where

$$g_R(x) = \sum_{i \sim j} \frac{1}{2} h_{R_{ij}}(\|x_i - x_j\| - d_{ij}) + \sum_i \sum_{k \in \mathcal{A}_i} \frac{1}{2} h_{R_{ik}}(\|x_i - a_k\| - r_{ik}). \quad (3)$$

This function is nonconvex and, in general, difficult to minimize. We shall provide a convex underestimator that tightly bounds each term of (3), thus leading to better estimation results than other relaxations which are not tight [17].

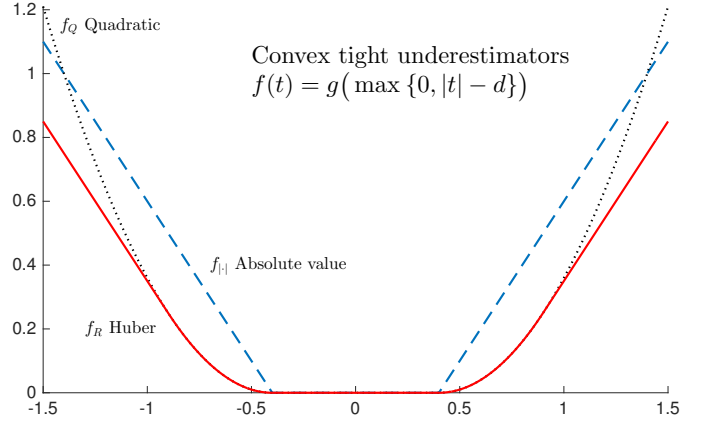


Fig. 2. All functions f are tight underestimators of the functions g in Figure 1. They are the convex envelopes and, thus, the best convex approximations to each one of the original nonconvex cost terms. The convexification is performed by restricting the arguments of g to be nonnegative.

III. CONVEX UNDERESTIMATOR

To convexify g_R we can replace each term by its convex hull¹, as depicted in Figure 2. Here, we observe that the high-power behavior is maintained, whereas the medium/low-power is only altered in the convexified area. We define the convex costs by composing any of the convex functions h with a nondecreasing function s

$$s(t) = \max\{0, t\}$$

which, in turn, operates on the discrepancies

$$\begin{aligned} \delta_{ij}(x) &= \|x_i - x_j\| - d_{ij}, \\ \delta_{ik}(x_i) &= \|x_i - a_k\| - r_{ik}. \end{aligned}$$

As $s(\delta_{ij}(x))$ and $s(\delta_{ik}(x))$ are nondecreasing and each one of the functions h is convex, then

$$f_R(x) = \sum_{i \sim j} \frac{1}{2} h_{R_{ij}}(s(\|x_i - x_j\| - d_{ij})) + \sum_i \sum_{k \in \mathcal{A}_i} \frac{1}{2} h_{R_{ik}}(s(\|x_i - a_k\| - r_{ik})) \quad (4)$$

is also convex. The cost function (4) also appears in Yousefi *et al.* [15] via a distinct reasoning. But the striking difference with respect to Yousefi *et al.* is how the cost (4) is exploited here to generate distributed solution methods where all nodes work in parallel, for the synchronous algorithm, or are randomly awoken, for the asynchronous algorithm.

A. Approximation quality of the convex underestimator

The quality of the convexified quadratic problem was addressed in [5], which we summarize here for the reader's convenience and extend to the two other convex problems.

The optimal value of the nonconvex g , denoted by g^* , is bounded by

$$f^* = f(x^*) \leq g^* \leq g(x^*),$$

¹The convex hull of a function γ , i.e., its best possible convex underestimator, is defined as $\text{conv } \gamma(x) = \sup \{\eta(x) : \eta \leq \gamma, \eta \text{ is convex}\}$. It is hard to determine in general [18].

TABLE I
BOUNDS ON THE OPTIMALITY GAP FOR THE EXAMPLE IN FIGURE 3

Cost	$g^* - f^*$	Eqs. (5)-(7)	Eqs. (8)-(10)
Quadratic	3.7019	5.5250	11.3405
Absolute value	1.1416	1.1533	3.0511
Robust Huber	0.1784	0.1822	0.4786

where x^* is the minimizer of the convex underestimator f , and

$$f^* = \min_x f(x),$$

is the minimum of function f . A bound for the optimality gap is, thus,

$$g^* - f^* \leq g(x^*) - f^*.$$

It is evident that in all cases (quadratic, Huber, and absolute value) f is equal to g when $\|x_i - x_j\| \geq d_{ij}$ and $\|x_i - a_k\| \geq r_{ik}$. When the function terms differ, say, for all edges² $i \sim j \in \mathcal{E}_2 \subset \mathcal{E}$, we have $s(\|x_i - x_j\| - d_{ij}) = 0$, leading to

$$g_Q^* - f_Q^* \leq \sum_{i \sim j \in \mathcal{E}_2} \frac{1}{2} (\|x_i^* - x_j^*\| - d_{ij})^2 \quad (5)$$

$$g_{|\cdot|}^* - f_{|\cdot|}^* \leq \sum_{i \sim j \in \mathcal{E}_2} \frac{1}{2} \|\|x_i^* - x_j^*\| - d_{ij}\| \quad (6)$$

$$g_R^* - f_R^* \leq \sum_{i \sim j \in \mathcal{E}_2} \frac{1}{2} h_{R_{ij}} (\|x_i^* - x_j^*\| - d_{ij}), \quad (7)$$

where

$$\mathcal{E}_2 = \{i \sim j \in \mathcal{E} : \|x_i^* - x_j^*\| < d_{ij}\}.$$

These bounds are an optimality gap guarantee available after the convexified problem is solved; they tell us how low our estimates can bring the original cost. Our bounds are tighter than the ones available *a priori* from applying [19, Th. 1], which are

$$g_Q^* - f_Q^* \leq \sum_{i \sim j} \frac{1}{2} d_{ij}^2 \quad (8)$$

$$g_{|\cdot|}^* - f_{|\cdot|}^* \leq \sum_{i \sim j} \frac{1}{2} d_{ij} \quad (9)$$

$$g_R^* - f_R^* \leq \sum_{i \sim j} \frac{1}{2} h_{R_{ij}} (d_{ij}). \quad (10)$$

For a single-node 1D example whose costs for a single noise realization are exemplified in Figure 3, the bounds in (5)-(7) and (8)-(10), averaged over 500 Monte Carlo trials, are presented in Table I. The true average gap $g^* - f^*$ is also listed. In the Monte Carlo trials we sampled a set of zero mean Gaussian random variables with $\sigma = 0.04$ for the baseline Gaussian noise and obtained a noisy range measurement as in (27) below. One of the measurements is then corrupted by a zero mean random variable with $\sigma = 4$, modelling outlier noise. These results show the tightness of the convexified function under such noisy conditions and also demonstrate the looseness of the *a priori* bounds in (8)-(10). We can observe in

²The same reasoning would apply to anchor terms.

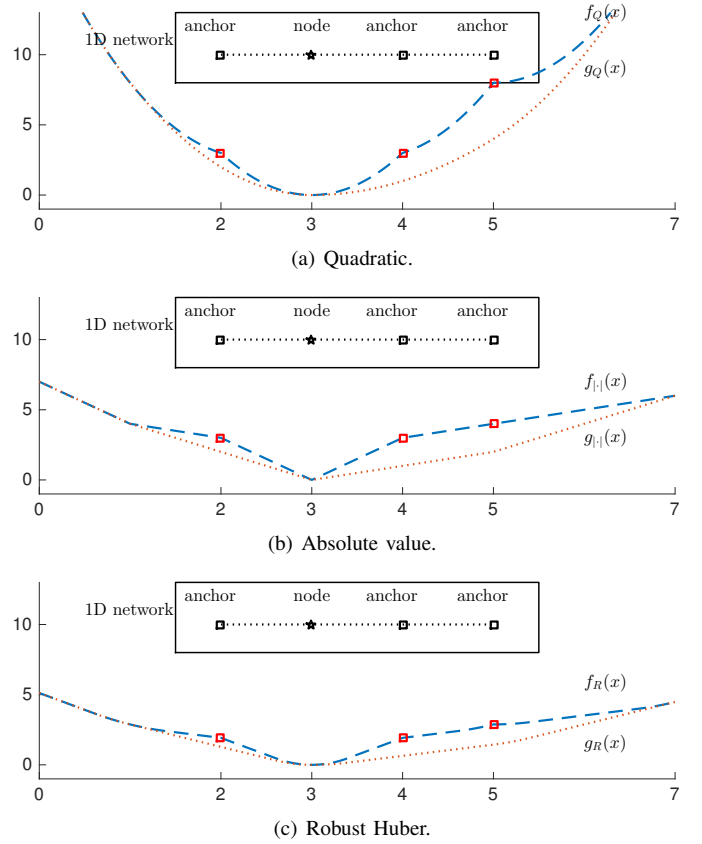


Fig. 3. Single node 1D example of the quality of the approximation of the true nonconvex costs $g(x)$ by the convexified functions $f(x)$. The node positioned at $x = 3$ has 3 neighboring anchors. The cost value is indicated in the vertical axis, while the tentative node position runs on the horizontal. The actual network is depicted above the plots.

Figure 3 why the Huber-based relaxation will perform better than the other two: not only do we use a robust dissimilarity, but we also add a smaller optimality gap with the surrogate. In the end, the Huber-based approximation will be tighter, thus conferring robustness to the estimator, as pointed out by Destino and Abreu [4].

IV. DISTRIBUTED AND ROBUST SENSOR NETWORK LOCALIZATION

We construct our algorithm by rewriting (4) as the infimum of a sum of Huber functions composed with a norm, and then by rewriting each of the terms with an alternative representation that uncovers the possibility of a naturally distributed, optimal method for the estimation of the unknown sensor positions. For the first step, we state the following:

Proposition 1. *Each term of the first summation of (4), corresponding to the edge $i \sim j$, has a variational representation*

$$h_{R_{ij}}(s(\|x_i - x_j\| - d_{ij})) = \inf_{\|y_{ij}\| \leq d_{ij}} h_{R_{ij}}(\|x_i - x_j - y_{ij}\|), \quad (11)$$

where $y_{ij} \in \mathbb{R}^p$ is an optimization variable.

The proof is detailed in Appendix A.

A. Alternative representation of the Huber function composed with a norm

Using the variational representation from Proposition 1, we can rewrite the convex unconstrained minimization Problem (4) as the constrained problem

$$\begin{aligned} & \underset{x,y,w}{\text{minimize}} \sum_{i \sim j} \frac{1}{2} h_{R_{ij}}(\|x_i - x_j - y_{ij}\|) + \\ & \quad \sum_i \sum_{k \in \mathcal{A}_i} \frac{1}{2} h_{R_{ik}}(\|x_i - a_k - w_{ik}\|) \quad (12) \\ & \text{subject to } \{\|y_{ij}\| \leq d_{ij}, i \sim j\} \\ & \quad \{\|w_{ik}\| \leq r_{ik}, i \in \mathcal{V}, k \in \mathcal{A}_i\} \end{aligned}$$

where $x = \{x_i : i \in \mathcal{V}\}$, $y = \{y_{ij} : i \sim j\}$, and $w = \{w_{ik} : i \in \mathcal{V}, k \in \mathcal{A}_i\}$. We put forward a new representation of the Huber function in (1), when composed with the norm of a vector as

$$\psi_R(u) = \|u\|^2 - d_R^2(u), \quad (13)$$

where we denote by $d_R^2(u)$ the squared distance of vector u to a ball of radius R centered at the origin. We use this representation to rewrite the cost in problem (12) as

$$\begin{aligned} & \sum_{i \sim j} \frac{1}{2} \|x_i - x_j - y_{ij}\|^2 - \frac{1}{2} d_{R_{ij}}^2(x_i - x_j - y_{ij}) + \\ & \sum_i \sum_{k \in \mathcal{A}_i} \frac{1}{2} \|x_i - a_k - w_{ik}\|^2 - \frac{1}{2} d_{R_{ik}}^2(x_i - a_k - w_{ik}), \end{aligned}$$

and we further work the problem, leading to

$$\begin{aligned} & \frac{1}{2} \|Ax - y\|^2 - \frac{1}{2} d_{\tilde{R}}^2(Ax - y) + \\ & \sum_{i \in \mathcal{V}} \frac{1}{2} \|x_i \otimes \mathbf{1} - \alpha_i - w_i\|^2 - \frac{1}{2} d_{\tilde{R}_a}^2(x_i \otimes \mathbf{1} - \alpha_i - w_i), \end{aligned}$$

where \tilde{R} is the Cartesian product of the balls $\{x : \|x\| \leq R_{ij}, i \sim j\}$, \tilde{R}_a is the Cartesian product of the balls $\{x : \|x\| \leq R_{ik}, k \in \mathcal{A}_i, i \in \mathcal{V}\}$, and $d_{\tilde{R}}^2(\cdot)$ is the squared distance to set \tilde{R} (similarly for $d_{\tilde{R}_a}^2(\cdot)$). Matrix $A = C \otimes I$ is the Kronecker product of the arc-node incidence matrix³ C associated with the graph \mathcal{G} , and the identity matrix with dimension of the ambient space (usually, 2 or 3). The terms $\alpha_i = \{a_k : k \in \mathcal{A}_i\}$ are the collections of the positions of the anchors within range of each node, and $w_i = \{w_{ik}, k \in \mathcal{A}_i\}$ are the collections of associated variables. Consider the aggregation of variables $z = (x, y, w)$; we define the constraint set in (12) as

$$\mathcal{Z} = \{(x, y, w) : \|y_{ij}\| \leq d_{ij}, i \sim j, \|w_{ik}\| \leq r_{ik}, k \in \mathcal{A}_i, i \in \mathcal{V}\}, \quad (14)$$

and the cost as

$$F(z) := \frac{1}{2} \|Bz\|^2 - \frac{1}{2} d_{\tilde{R}}^2(Bz) + \frac{1}{2} \|Ez - \alpha\|^2 - \frac{1}{2} d_{\tilde{R}_a}^2(Ez - \alpha), \quad (15)$$

³The arc-node incidence matrix C is a $|\mathcal{E}| \times |\mathcal{V}|$ matrix. The rows and the columns of C are indexed by \mathcal{E} and \mathcal{V} , respectively. The (e, i) -entry of C is 0 if node i and edge e are not incident, and otherwise it is 1 or -1 according to the direction agreed on the operation onset by the two nodes involved in edge e .

where $B = [A \quad -I \quad 0]$, matrix $E = [E_x \quad 0 \quad -I]$, and E_x is a selector matrix of the anchor terms associated with each separate node. With this notation, (12) becomes

$$\begin{aligned} & \text{minimize} \quad F(z) \\ & \text{subject to} \quad z \in \mathcal{Z}. \end{aligned} \quad (16)$$

B. Gradient

To compute the gradient of the cost in (15), we need the gradient of the squared distance to a convex set — a result from convex analysis (see [18, Prop. X.3.2.2, Th. X.3.2.3]). Let us denote the squared distance to the convex set C as

$$\phi(u) = \frac{1}{2} d_C^2(u).$$

Then, from convex analysis, we know that ϕ is convex, differentiable, and its gradient is

$$\nabla \phi(u) = u - P_C(u), \quad (17)$$

where $P_C(u)$ is the orthogonal projection of point u onto the set C ,

$$P_C(u) = \underset{y \in C}{\operatorname{argmin}} \|u - y\|.$$

Knowing this, we can compute the gradient of (13) as

$$\nabla \psi_R(u) = 2P_R(u)$$

and the gradient of (15) as

$$\begin{aligned} \nabla F(z) &= \frac{1}{2} B^\top \nabla \psi_{\tilde{R}}(Bz) + \frac{1}{2} E^\top \nabla \psi_{\tilde{R}_a}(Ez - \alpha) \\ &= B^\top P_{\tilde{R}}(Bz) + E^\top P_{\tilde{R}_a}(Ez - \alpha). \end{aligned} \quad (18)$$

C. Lipschitz constant

It is widely known that projections onto convex sets shrink distances [20], i.e.,

$$\|P_C(u) - P_C(v)\| \leq \|u - v\|,$$

and this means that the gradient of (13) is Lipschitz continuous with Lipschitz constant 1. Using this, we can compute a Lipschitz constant for (18). First, let us focus on the inter-node term in (18):

$$\begin{aligned} \|B^\top P_{\tilde{R}}(Bu) - B^\top P_{\tilde{R}}(Bv)\| &\leq \|B\| \|P_{\tilde{R}}(Bu) - P_{\tilde{R}}(Bv)\| \\ &\leq \|B\| \|Bu - Bv\| \\ &\leq \|B\|^2 \|u - v\| \\ &= \lambda_{\max}(BB^\top) \|u - v\|. \end{aligned}$$

The maximum eigenvalue of BB^\top can be bounded by

$$\begin{aligned} \lambda_{\max}(BB^\top) &= \lambda_{\max} \left(\begin{bmatrix} A & -I & 0 \end{bmatrix} \begin{bmatrix} A^\top \\ -I \\ 0 \end{bmatrix} \right) \\ &= \lambda_{\max}(AA^\top + I) \\ &= (1 + \lambda_{\max}(AA^\top)) \\ &= (1 + \lambda_{\max}(L)) \\ &\leq (1 + 2\delta_{\max}), \end{aligned} \quad (19)$$

where L is the graph Laplacian, and δ_{\max} is the maximum node degree of the network. A proof of the inequality $\lambda_{\max} \leq 2\delta_{\max}$ is in Bapat [21]. In the same way, for the node-anchor terms, we have

$$\begin{aligned} \|E^\top P_{\tilde{R}a}(Eu) - E^\top P_{\tilde{R}a}(Ev)\| &\leq \|E\|^2 \|u - v\| \\ &= \lambda_{\max}(EE^\top) \|u - v\| \end{aligned}$$

and this constant can be upper-bounded by

$$\begin{aligned} \lambda_{\max}(EE^\top) &= \lambda_{\max} \left(\begin{bmatrix} E_x & 0 & -I \end{bmatrix} \begin{bmatrix} E_x^\top \\ 0 \\ -I \end{bmatrix} \right) \\ &= \lambda_{\max}(E_x E_x^\top + I) \\ &\leq (1 + \lambda_{\max}(E_x E_x^\top)) \\ &\leq \left(1 + \max_{i \in \mathcal{V}} |\mathcal{A}_i|\right). \end{aligned} \quad (20)$$

From (19) and (20) we can see that a Lipschitz constant for (15) is

$$L_F = 2 + 2\delta_{\max} + \max_{i \in \mathcal{V}} |\mathcal{A}_i|. \quad (21)$$

We stress that this constant is small, does not depend on the size of the network, and can be computed in a distributed way [22].

D. Synchronous algorithm

The gradient in (18) and its Lipschitz continuity, with the constant in (21), equip us to use the optimal gradient optimization method due to Nesterov ([23] [24]), further developed by Beck and Teboulle [25]. Firstly, we must write the problem as an unconstrained minimization using an indicator function $I_{\mathcal{Z}}(u) = \begin{cases} 0 & \text{if } u \in \mathcal{Z} \\ +\infty & \text{otherwise} \end{cases}$, and incorporate

the constraints in the problem formulation. Then we perform the proximal minimization of the unconstrained problem. The result for our reformulation is shown in Algorithm 1. Here, \mathcal{Y}_{ij} and \mathcal{W}_{ik} are the sets $\{x : \|x\| \leq d_{ij}\}$, and $\{x : \|x\| \leq r_{ik}\}$, respectively. Also, \mathcal{Y} and \mathcal{W} are the constraint sets associated with the acquired measurements between sensors, and between anchors and sensors, respectively, and N_i is the set of neighbor nodes of node i . We denote the entries of ∇F regarding variable x_i as G . We observe that each block of $z = (x, y, w)$ at iteration t will only need local neighborhood information, as shown in Algorithm 1. To demonstrate the natural distribution of the method we go back to (18). Here, the term $E^\top P_{\tilde{R}a}(Ez - a)$ only involves anchor measurements relative to each node, and so it is distributed. The term $B^\top P_{\tilde{R}}(Bz)$ is less clear. The vector Bz collects $x_i - x_j - y_{ij}$ for all edges $i \sim j$ and to it we apply the projection operator onto the Cartesian product of balls. This is the same as applying a projection of each edge onto each ball. When left multiplying with B^\top we get $B^\top P_{\tilde{R}}(Bz)$. The left multiplication by B^\top will group at the position of each node variable x_i the contributions of all incident edges to node i . To update the y_{ij} variables we could designate one of the incident nodes as responsible for the update and then communicate

Algorithm 1 Synchronous method: syncHuber

Input: $L_F; \{d_{ij} : i \sim j \in \mathcal{E}\}; \{r_{ik} : i \in \mathcal{V}, k \in \mathcal{A}\};$

Output: \hat{x}

```

1: each node  $i$  chooses arbitrary  $x_i^0 = x_i^{-1}$ ;
2: set  $y_{ij}^0 = P_{\mathcal{Y}_{ij}}(x_i^0 - x_j^0)$ ,  $\mathcal{Y}_{ij} = \{y \in \mathbb{R}^p : \|y\| \leq d_{ij}\}$ ;
   and  $w_{ik}^0 = P_{\mathcal{W}_{ik}}(x_i^0 - a_k)$ ,  $\mathcal{W}_{ik} = \{w \in \mathbb{R}^p : \|w\| \leq r_{ik}\}$ ;
3:  $t = 0$ ;
4: while some stopping criterion is not met, each node  $i$  do
5:    $t = t + 1$ ;
6:    $\xi_i = x_i^{t-1} + \frac{t-2}{t+1} (x_i^{t-1} - x_i^{t-2})$ ;
7:   broadcast  $\xi_i$  to all neighbors;
8:   for all  $j$  in the neighbor set  $N_i$  do
9:      $v_{ij} = y_{ij}^{t-1} + \frac{t-2}{t+1} (y_{ij}^{t-1} - y_{ij}^{t-2})$ ;
10:     $y_{ij}^t = P_{\mathcal{Y}_{ij}}(v_{ij} + \frac{1}{L_F} P_{R_{ij}}(\xi_i - \xi_j - v_{ij}))$ ;
11:   end for
12:   for all  $k$  in the anchor set  $\mathcal{A}_i$  do
13:      $\omega_{ik} = w_{ik}^{t-1} + \frac{t-2}{t+1} (w_{ik}^{t-1} - w_{ik}^{t-2})$ ;
14:      $w_{ik}^t = P_{\mathcal{W}_{ik}}(\omega_{ik} + \frac{1}{L_F} P_{Ra_{ik}}(\xi_i - a_{ik} - \omega_{ik}))$ ;
15:   end for
16:
17:    $x_i^t = \xi_i - \frac{1}{L_F} G$ ;
18: end while
19: return  $\hat{x}_i = x_i^t$ 

```

the result to the non-computing neighbor. But, to avoid this expensive extra communication, we decide that each node i should compute its own y_{ij} , where $y_{ij} = -y_{ji}$ for all edges. Also, with this device, the gradient entry regarding variable x_i would be $\sum_{j \in N_i} C_{(i \sim j, i)} P_{R_{ij}}(C_{(i \sim j, i)}(x_i - x_j - y_{ij}))$. The symbol $C_{(i \sim j, i)}$ denotes the arc-node incidence matrix entry relative to edge $i \sim j$ (row index) and node i (column index). As the projection onto a ball of radius R centered at the origin can be written as $P_R(u) = \begin{cases} \frac{u}{\|u\|} R & \text{if } \|u\| > R \\ u & \text{if } \|u\| \leq R \end{cases}$, then $P_R(-u) = -P_R(u)$, and, thus, the gradient entry regarding variable x_i becomes $\sum_{j \in N_i} P_{R_{ij}}(x_i - x_j - y_{ij})$, as stated in Algorithm 1. Each node i will update the current estimate of its own position, each one of the y_{ij} for all the incident edges and the anchor terms w_{ik} , if any. In step 6 we have the extrapolation step for each x_i , whereas in steps 9 and 13 we can see the update of the extrapolation steps for each one of the edge variables y_{ij} , and w_{ik} , respectively.

E. Asynchronous algorithm

In Section IV-D we presented a distributed method addressing the robust network localization problem in a scalable manner, where each node uses information from its neighborhood and performs a set of simple arithmetic computations. But

Algorithm 2 Asynchronous method: asyncHuber

Input: $L_F; \{d_{ij} : i \sim j \in \mathcal{E}\}; \{r_{ik} : i \in \mathcal{V}, k \in \mathcal{A}\};$
Output: \hat{x}

- 1: each node i chooses random $x_i(0)$;
 - 2: set $y_{ij}^0 = P_{\mathcal{Y}_{ij}}(x_i^0 - x_j^0)$, $\mathcal{Y}_{ij} = \{y \in \mathbb{R}^p : \|y\| \leq d_{ij}\}$
and $w_{ik}^0 = P_{\mathcal{W}_{ik}}(x_i^0 - a_k)$, $\mathcal{W}_{ik} = \{w \in \mathbb{R}^p : \|w\| \leq r_{ik}\}$
 - 3: $t = 0$;
 - 4: **while** some stopping criterion is not met, each node i **do**
 - 5: $t = t + 1$;
 - 6: $x_i^t = \begin{cases} \underset{\xi_i, \{v_{ij} \in \mathcal{Y}_{ij}, i \sim j\}, \{\omega_{ik} \in \mathcal{W}_{ik}, k \in \mathcal{A}_k\}}{\operatorname{argmin}} F_i(\xi_i, \{v_{ij}\}, \{\omega_{ik}\}) & \text{if } \chi_t = i \\ x_i^{t-1} & \text{otherwise;} \end{cases}$
 - 7: if $\xi_t = i$, broadcast x_i^t to neighbors
 - 8: **end while**
 - 9: **return** $\hat{x} = x^t$
-

the results still depend critically on synchronous computation, where nodes progress in lockstep through iterations. As the number of processing nodes becomes very large, this synchronization can become seriously difficult — and unproductive. An asynchronous approach is called for in such very large-scale and faulty settings. In an asynchronous time model, the nodes move forward independently and algorithms withstand certain types of faults, like temporary unavailability of a node. To address this issue, we present a fully asynchronous method, based on a broadcast gossip scheme (c.f. Shah [26] for an extended survey of gossip algorithms).

Nodes are equipped with independent clocks ticking at random times (say, as Poisson point processes). When node i 's clock ticks, it performs the update of its variables and broadcasts the update to its neighbors. Let the order of node activation be collected in $\{\chi_t\}_{t \in \mathbb{N}}$, a sequence of independent random variables taking values on the set \mathcal{V} , such that

$$\mathbb{P}(\chi_t = i) = P_i > 0. \quad (22)$$

Then, the asynchronous update of variables on node i can be described as in Algorithm 2. To compute the minimizer in step 6 of Algorithm 2 it is useful to recast Problem (16) as

$$\begin{aligned} \underset{x, y, w}{\text{minimize}} \quad & \sum_i \left(\sum_{j \in N_i} \frac{1}{4} \|x_i - x_j - y_{ij}\|^2 - \frac{1}{4} d_{R_{ij}}^2(x_i - x_j - y_{ij}) + \right. \\ & \left. \sum_{k \in \mathcal{A}_i} \frac{1}{2} \|x_i - a_k - w_{ik}\|^2 - \frac{1}{2} d_{R_{ik}}^2(x_i - a_k - w_{ik}) \right) \\ \text{subject to} \quad & y \in \mathcal{Y} \\ & w \in \mathcal{W}, \end{aligned} \quad (23)$$

where the factor $\frac{1}{4}$ accounts for the duplicate terms when considering summations over nodes instead of over edges. By fixing the neighbor positions, each node solves a single source

localization problem; this setup leads to

$$\begin{aligned} \underset{x_i, y_{ij}, w_{ik}}{\text{minimize}} \quad & F_i(x_i, \{y_{ij}, i \sim j\}, \{w_{ik}, k \in \mathcal{A}_i\}) \\ \text{subject to} \quad & \|y_{ij}\| \leq d_{ij} \\ & \|w_{ik}\| \leq r_{ik}, \end{aligned} \quad (24)$$

where

$$\begin{aligned} F_i(x_i, \{y_{ij}, i \sim j\}, \{w_{ik}, k \in \mathcal{A}_i\}) = & \sum_{j \in N_i} \frac{1}{4} \|x_i - x_j - y_{ij}\|^2 - \frac{1}{4} d_{R_{ij}}^2(x_i - x_j - y_{ij}) + \\ & \sum_{k \in \mathcal{A}_i} \frac{1}{2} \|x_i - a_k - w_{ik}\|^2 - \frac{1}{2} d_{R_{ik}}^2(x_i - a_k - w_{ik}). \end{aligned} \quad (25)$$

Problem (24) is convex, solvable at each node by a general purpose solver. Nevertheless, that approach would not take advantage of the specific problem structure, thus depriving the solution of an efficient and simpler computational procedure. Again, Problem (24) can be solved by the Nesterov optimal first order method, because the gradient of F_i is Lipschitz continuous in $x_i, \{y_{ij}\}$, and $\{w_{ik}\}$, accepting the same Lipschitz constant as F , in (21).

V. CONVERGENCE ANALYSIS

In this section we address the convergence of Algorithms 1 and 2. We provide convergence guarantees and rate of convergence for the synchronous version, and we also prove convergence for the asynchronous method.

A. Synchronous algorithm

As shown in Section IV, Problem (16) is convex and the cost function has a Lipschitz continuous gradient. As proven by Nesterov ([23], [24]), and further developed by Beck and Teboulle [25], Algorithm 1 converges at the optimal rate $O(t^{-2})$; specifically, $F(z^t) - F^* \leq \frac{2L_F}{(t+1)^2} \|z^0 - z^*\|^2$, where F^* is the optimal value and z^* is a minimizer of Problem (16).

B. Asynchronous algorithm

To investigate the convergence of Algorithm 2, we need the following assumptions:

Assumption 2. *The topology of the network conforms to:*

- *The graph \mathcal{G} is connected;*
- *There is at least one node in \mathcal{G} with an anchor measurement.*

These assumptions are naturally fulfilled in the network localization problem: if the network is supporting several disconnected components, then each can be treated as a different network, and, for disambiguation, localization requires the availability of 3 anchors in 2D and 4 anchors in 3D.

The convergence of Algorithm 2 is stated next.

Theorem 3 (Almost sure convergence). *Let Assumption 2 hold. Consider Problem (16), and the sequence $\{z^t\}_{t \in \mathbb{N}}$ generated by Algorithm 2. Define the solution set as $\mathcal{Z}^* = \{z \in \mathcal{Z} : F(z) = F^*\}$. Then*

- 1) $d_{\mathcal{Z}^*}(z^t) \rightarrow 0$, a.s.;
- 2) $F(z^t) \rightarrow F^*$, a.s.

The reader can find the proof in Appendix B. It is also possible to state that, with probability one, Algorithm 2 converges in a *finite* number of iterations. The result is stated in the following theorem, also proven in Appendix B.

Theorem 4. *For a prescribed precision ϵ , the sequence of iterates $\{z^t\}_{t \in \mathbb{N}}$ converges in K_ϵ iterations. The expected value of this number of iterations is*

$$\mathbb{E}[K_\epsilon] \leq \frac{F(z^0) - F^*}{b_\epsilon}, \quad (26)$$

where b_ϵ is a constant that depends on the specified ϵ .

VI. NUMERICAL EXPERIMENTS

A. Underestimator performance

We assess the performance of the three considered loss functions through simulation. The experimental setup consists in a uniquely localizable geometric network deployed in a square area with side of 1Km, with four anchors (blue squares in Figure 4) located at the corners, and ten sensors, (red stars). Measurements are also visible as dotted green lines. The average node degree⁴ of the network is 4.3. The regular noisy range measurements are generated according to

$$\begin{aligned} d_{ij} &= ||x_i^* - x_j^*|| + \nu_{ij}, \\ r_{ik} &= ||x_i^* - a_k|| + \nu_{ik}, \end{aligned} \quad (27)$$

where x_i^* is the true position of node i , and $\{\nu_{ij} : i \sim j \in \mathcal{E}\} \cup \{\nu_{ik} : i \in \mathcal{V}, k \in \mathcal{A}_i\}$ are independent Gaussian random variables with zero mean and standard deviation 0.04, corresponding to an uncertainty of about 40m. Node 7 is malfunctioning and all measurements related to it are corrupted by Gaussian noise with standard deviation 4, corresponding to an uncertainty of 4Km. The convex optimization problems were solved with `cvx` [27]. We ran 100 Monte Carlo trials, sampling both regular and outlier noise.

The performance metric used to assess accuracy is the positioning error per sensor defined as

$$\epsilon(m) = \frac{||\hat{x}(m) - x^*||}{|\mathcal{V}|}, \quad (28)$$

where $\hat{x}(m)$ corresponds to the position estimates for all sensors in Monte Carlo trial m . The empirical mean of the positioning error is defined as

$$\epsilon = \frac{1}{M} \sum_{m=1}^M \epsilon(m), \quad (29)$$

where M is the number of Monte Carlo trials. In Figure 4 we can observe that clouds of estimates from g_R and g_Q gather around the true positions, except for the malfunctioning node 7. Note the increased spread of blue dots around nodes with edges connecting to node 7, indicating that g_R better preserves the nodes' ability to localize themselves, despite

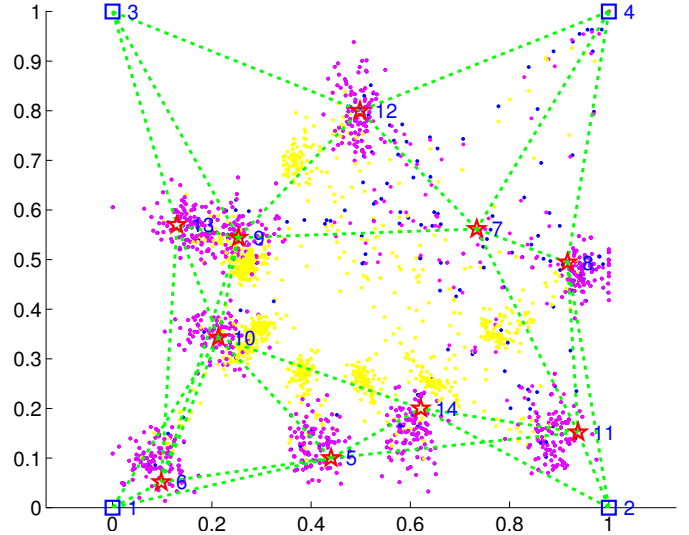


Fig. 4. Underestimator performance: Estimates of sensor positions for the three loss functions; We plotted in yellow the monte carlo results of minimizing f_{L1} , the L_1 loss; in blue we can see the estimates resulting from minimizing f_Q , the quadratic loss; in the same way, magenta dots represent the output for function f_R , the Huber loss. It is noticeable that the L_1 loss is not able to correctly estimate positions whose measurements are corrupted with Gaussian noise. The perturbation in node 7 has more impact in the dispersion of blue dots than magenta dots around its neighbors.

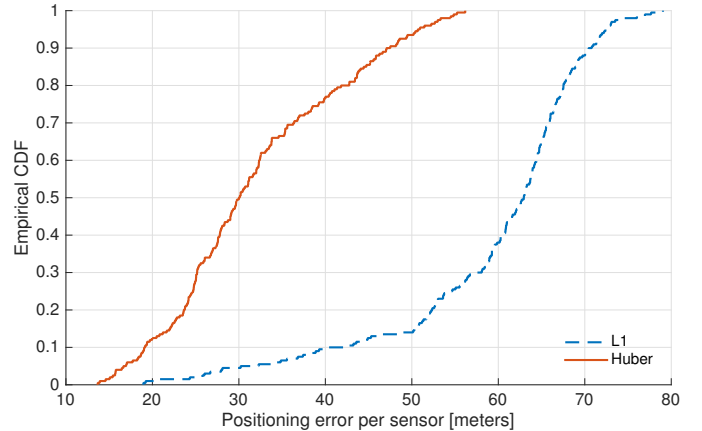


Fig. 5. Underestimator performance: Empirical CDF for the positioning error per sensor, in meters, for the Gaussian outlier noise experiment.

their confusing neighbor, node 7. This intuition is confirmed by the empirical CDFs of estimation errors shown in Figure 5, which demonstrate that the Huber robust cost can reduce the error per sensor by an average of 28.5 meters, when compared with the L_1 discrepancy. Also, as expected, the malfunctioning node cannot be positioned by any of the algorithms. The sensitivity to the value of the Huber parameter R in (1) is only moderate, as shown in Figure 6. In fact, the error per sensor of the proposed estimator is always the smallest for all tested values of the parameter. We observe that the error increases when R approaches the standard deviation of the regular Gaussian noise, meaning that the Huber loss gets closer

⁴To characterize the network we use the concepts of *node degree* k_i , which is the number of edges connected to node i , and *average node degree* $\langle k \rangle = 1/n \sum_{i=1}^n k_i$.

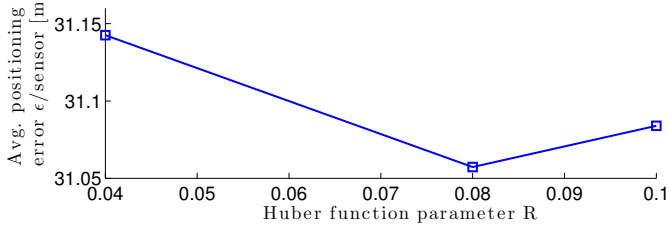


Fig. 6. Underestimator performance: Average positioning error versus the value of the Huber function parameter R . Accuracy is maintained for a wide range of parameter values.

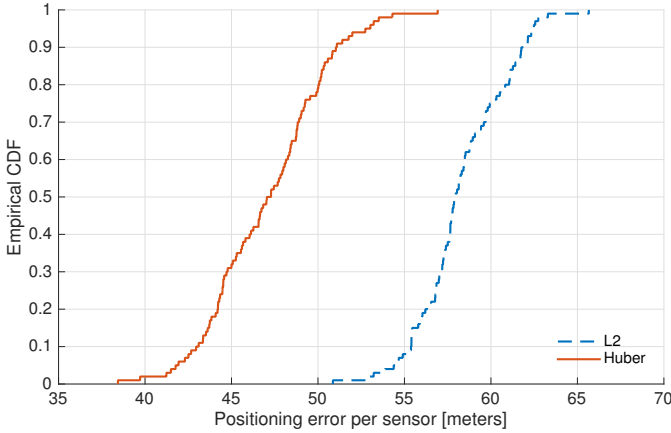


Fig. 7. Underestimator performance: Empirical CDF for the positioning error per sensor, in meters, for the biased node experiment.

to the L_1 loss and, thus, is no longer adapted to the regular noise ($R = 0$ corresponds exactly to the L_1 loss); in the same way, as R increases, so does the quadratic section, and the estimator gets less robust to outliers, so, again, the error increases.

Another interesting experiment is to see what happens when the faulty sensor produces measurements with consistent errors or bias. We ran 100 Monte Carlo trials in the same setting, but node 7 measurements are now consistently 10% of the real distance to each neighbor. The empirical CDF for the positioning error per sensor is shown in Figure 7. Here we observe a significant performance gap between the alternative costs — in average about 11.2 meters — so the Huber formulation proves to be superior even with biased sensors.

B. Performance of the distributed synchronous Algorithm 1

We tested Algorithm 1 using the same setup as in the previous section, with node 7 contaminated with added Gaussian noise with standard deviation of 4. We benchmark our method comparing with the performance of the centralized solutions in Oğuz-Ekim *et al.*, [1], which we denote as “Median” below, the SDP presented by Simonetto and Leus⁵ [17], and also the distributed locally convergent algorithm by Korkmaz and

⁵In [17], the authors present a distributed ESDP algorithm which is a relaxation of the centralized SDP. As the simulation time for the distributed, edge-based algorithm is considerable we benchmarked against the tighter and more accurate centralized SDP solution.

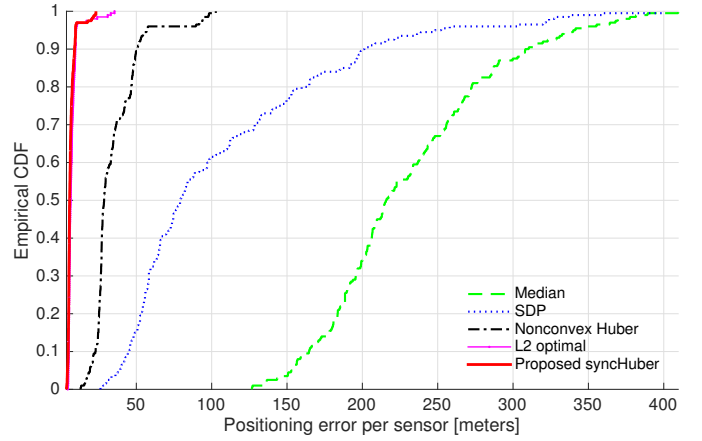


Fig. 8. Accuracy of the distributed synchronous algorithm: Empirical CDF for the positioning error per sensor, in meters, for the Gaussian outlier noise experiment, discarding the positioning error of the malfunctioning node.

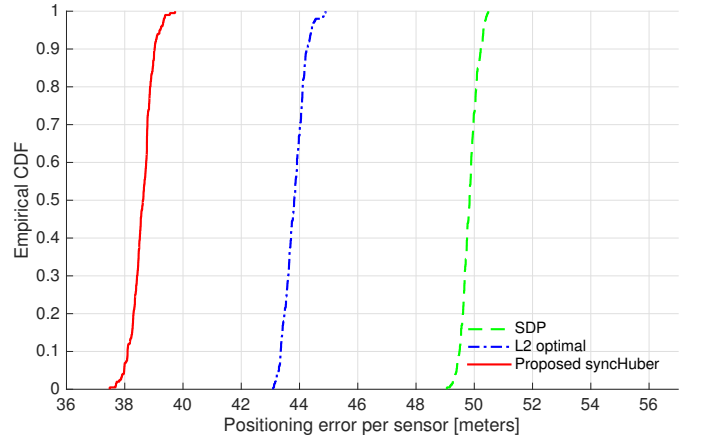


Fig. 9. Accuracy of the distributed synchronous algorithm: Empirical CDF for the positioning error per sensor, in meters, for the biased node experiment discarding the positioning error of the malfunctioning node.

Van der Veen⁶ [13]. The results are summarized in Figure 8. Here the empirical CDFs of the positioning error (28) show a superior accuracy of our syncHuber algorithm.

When analyzing the results for the biased experiment as described in the previous section, it is noticeable that the syncHuber algorithm beats the state-of-the-art [5] for the quadratic discrepancy by more than 5 meters per sensor in average positioning error, as depicted in Figure 9. As expected from the results in the previous section, when we compare to a L_1 -type algorithm — in this case the “Median” from Oğuz-Ekim *et al.* [1] — the improvement of performance of our solution is outstanding (on average about 120 meters per sensor), as depicted in Figure 10.

C. Performance of the distributed asynchronous Algorithm 2

Here, we tested Algorithm 2, asyncHuber, using the same setup as in the previous sections, with node 7 contaminated with added Gaussian noise with standard deviation of 4. We

⁶This distributed method attacks directly the nonconvex cost (3), thus delivering a local solution, that depends on the initialization point. The algorithm was initialized with Gaussian noise.

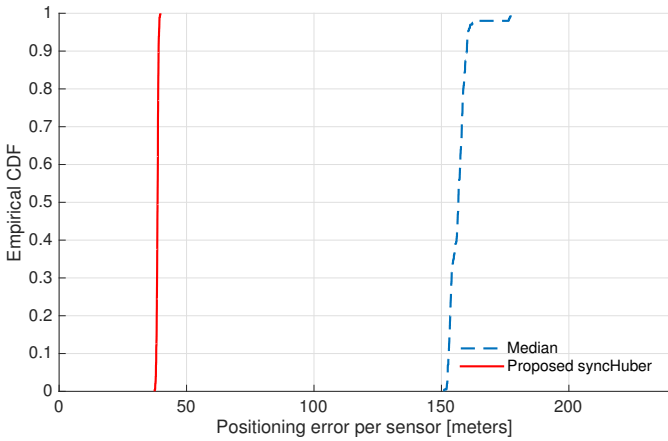


Fig. 10. Accuracy of the distributed synchronous algorithm: Empirical CDF for the positioning error per sensor, in meters, for the biased experiment discarding the positioning error of the malfunctioning node.

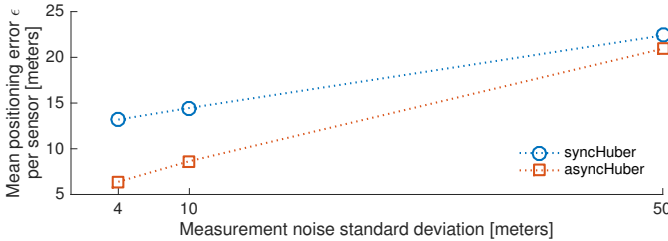


Fig. 11. Accuracy of the distributed asynchronous algorithm: Mean positioning error of synchronous algorithm 1 versus asynchronous algorithm 2, discarding the positioning error of the malfunctioning node. Both algorithms were run with the same communication load.

benchmarked it against the synchronous Algorithm 1, syncHuber, since both minimize the same cost function. The algorithms were allowed to run with the same communication load. The mean positioning error for the considered noise levels is depicted in Figure 11. We can observe that the asyncHuber algorithm fares better than syncHuber for the same communication amount. This is an interesting phenomenon empirically observed in different optimization algorithms when comparing deterministic and randomized versions. In fact, Bertsekas and Tsitsiklis [28, Section 6.3.5] provide a proof of this behavior for a restricted class of algorithms. Figure 12 further explores the experimental data, by examining the CDF of the positioning error for the tested Monte Carlo trials. Here we see the superior accuracy of the asynchronous Huber Algorithm 2, for the same communications volume. We must, nevertheless, emphasize that this result does not correspond to a faster algorithm, in terms of running time: syncHuber in one iteration updates all of the nodes positions in parallel, and broadcasts the current estimates across neighbors, whereas in asyncHuber only one node operates at a time. As the wireless medium might be much more intensively used for synchronous updates than for random gossip interactions, it seems entirely possible that for the same operation time syncHuber will outperform asyncHuber — at the expense of greater overall power consumption.

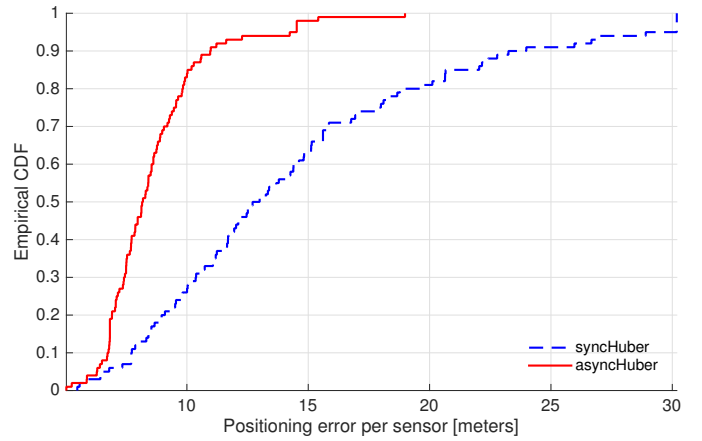


Fig. 12. Accuracy of the distributed asynchronous algorithm: CDF of the positioning error of synchronous algorithm 1 versus asynchronous algorithm 2, discarding the positioning error of the malfunctioning node. Both algorithms were run with the same communication load. Experiment with measurements contaminated by medium power noise ($\sigma = 0.01$), corresponding to 10 meters of standard deviation for a square with 1 Km sides.

VII. DISCUSSION AND CONCLUSIONS

We presented two distributed, fast, and robust localization algorithms that take noisy ranges and a few anchor locations, and output accurate estimates of the node positions. We approximated the difficult, nonconvex problem based on the Huber discrepancy in (2) with a convex envelope of terms, robust to outliers. How does the Huber-based approximation in (4) compares with similar L_1 and L_2 underestimators, frequent in robust estimation contexts? A smaller optimality gap means a more robust approximation [4]: We designed a bound that certifies the gap between the nonconvex and surrogate optimal values for Huber, L_1 and L_2 and shows a tighter gap in the Huber case. A numerical analysis of a star network in 1D unveiled that the optimality gap for the Huber approximation was one order of magnitude less than the quadratic or absolute value convexified problems, with respect to their nonconvex counterparts. Numerical network localization trials verify the surrogate robust behavior under different types of outlier noise. In order to develop a distributed method we needed to transform our cost. So, we proposed a new representation of the Huber function composed with a norm, and arrived at a novel distributed gradient method, syncHuber, with optimal convergence rate. But our syncHuber algorithm requires synchronization, which is a difficult demand for many applications. Thus, we put forward a novel asynchronous method for robust network localization, asyncHuber, converging with probability one. Nevertheless, like any other relaxation method, ours are prone to the anchor convex hull problem: preliminary results show the positioning accuracy degrades — albeit graciously — when nodes' positions depart from the anchor convex hull. Arguably, this is not a big issue because engineers in general can control the choice or placement of anchoring landmarks, and can delimit the area under survey.

In sum, both our algorithms work with simple computations at each node and minimal communication payloads, have provable convergence and show a superior performance in our

numerical experiments. Also, they do not require knowledge of a Hamiltonian path in the network, which simplifies real-world implementation, unlike the method presented by Yousefi *et al.* [15]. In average, the positioning error of Algorithm 1 is less 120m for a deployment in a square of 1Km sides than the state-of-the-art L_1 centralized method for robust network localization of Oğuz-Ekim *et al.* [1].

ACKNOWLEDGMENT

The authors would like to thank Pinar Oğuz-Ekim and Andrea Simonetto for providing the MATLAB implementations of their published algorithms. Also, we thank João Xavier for the interesting discussions during this research.

APPENDIX A PROOF OF PROPOSITION 1

We define a function $\phi(u) = \max\{0, h_{R_{ij}}(u)\}$, and restate a generic term of the first summation in (4)

$$h_{R_{ij}}(s(\|x_i - x_j\| - d_{ij}))$$

as

$$\phi(s(\|x_i - x_j\| - d_{ij})),$$

which represents the same mathematical object, because $s(\|x_i - x_j\| - d_{ij})$ is always nonnegative. Now, we prove the equivalence relation (11), beginning by

$$\phi(s(\|x_i - x_j\| - d_{ij})) \leq \inf_{\|y_{ij}\| \leq d_{ij}} \phi(\|x_i - x_j - y_{ij}\|). \quad (30)$$

We choose any $\bar{y}_{ij} : \|\bar{y}_{ij}\| \leq d_{ij}$, and note that

$$\begin{aligned} s(\|x_i - x_j\| - d_{ij}) &= \inf_{\|y_{ij}\| \leq d_{ij}} \|x_i - x_j - y_{ij}\| \\ &\leq \|x_i - x_j - \bar{y}_{ij}\|. \end{aligned}$$

As ϕ is nondecreasing,

$$\phi(s(\|x_i - x_j\| - d_{ij})) \leq \phi(\|x_i - x_j - \bar{y}_{ij}\|),$$

for all \bar{y}_{ij} , and in particular,

$$\phi(s(\|x_i - x_j\| - d_{ij})) \leq \inf_{\|y_{ij}\| \leq d_{ij}} \phi(\|x_i - x_j - y_{ij}\|),$$

which proves (30). We now establish

$$\phi(s(\|x_i - x_j\| - d_{ij})) \geq \inf_{\|y_{ij}\| \leq d_{ij}} \phi(\|x_i - x_j - y_{ij}\|). \quad (31)$$

We choose y_{ij}^* , a minimizer of the optimization problem in the RHS of (31). We know that

$$s(\|x_i - x_j\| - d_{ij}) = \|x_i - x_j - y_{ij}^*\|$$

and so

$$\phi(s(\|x_i - x_j\| - d_{ij})) = \phi(\|x_i - x_j - y_{ij}^*\|),$$

and as ϕ is monotonic,

$$\phi(\|x_i - x_j - y_{ij}^*\|) \leq \phi(\|x_i - x_j - y_{ij}\|),$$

for all $y_{ij} : \|y_{ij}\| \leq d_{ij}$. In particular,

$$\phi(\|x_i - x_j - y_{ij}^*\|) \leq \inf_{\|y_{ij}\| \leq d_{ij}} \phi(\|x_i - x_j - y_{ij}\|),$$

which proves (31), and concludes the proof of Proposition 1.

APPENDIX B PROOFS OF THEOREMS 3 AND 4

A. Definitions

First, we review the definition of a block optimal point and describe some useful mathematical objects used on the proofs.

Definition 5. A point $z^\bullet = (z_i^\bullet)_{i \in \mathcal{V}}$ is block optimal for the function F in (15) if, for all i , $z_i^\bullet \in \operatorname{argmin}_{w_i \in \mathcal{Z}_i} F(z_1^\bullet, \dots, w_i, \dots, z_n^\bullet)$ [29].

We define the sets

$$\mathcal{Z}^\star = \{z \in \mathcal{Z} : F(z) = F^\star\} \quad (32)$$

$$\mathcal{Z}_\epsilon = \{z : d_{\mathcal{Z}^\star}(z) < \epsilon\} \quad (33)$$

$$\mathcal{Z}_\epsilon^c = \{z : d_{\mathcal{Z}^\star}(z) \geq \epsilon\} \quad (34)$$

$$\mathcal{Z}_F = \{z : F(z) \leq F(z^0)\} \quad (35)$$

$$\hat{\mathcal{Z}}_\epsilon^c = \mathcal{Z}_F \cap \mathcal{Z}_\epsilon^c, \quad (36)$$

where $\hat{\mathcal{Z}}_\epsilon^c$ is the set of all points whose distance to the optimal set \mathcal{Z}^\star is larger than ϵ , but also belong to the sublevel set of F . We will see that the iterates of Algorithm 2 will belong to $\hat{\mathcal{Z}}_\epsilon^c$ until they reach the absorbing set \mathcal{Z}_ϵ . We also define the *expected improvement function* as

$$\psi(z) = \mathbb{E}[F(Z(k+1)) | Z(k) = z] - F(z), \quad (37)$$

and the coordinate optimal function as

$$F^i(z) = \min_{w_i \in \mathcal{Z}_i} F(z_1, \dots, w_i, \dots, z_n). \quad (38)$$

It is easy to see that the expected improvement ψ can also be written as

$$\psi(z) = \sum_{i=1}^n (F^i(z) - F(z)) P_i, \quad (39)$$

where P_i is the probability of the event “node i is awoken at time t ” (we recall the independence of the random variables χ_t defined in (22)). For notational convenience, we introduce the function

$$\phi(z) = -\psi(z), \quad (40)$$

which, by construction of Algorithm 2, is always non-negative.

1) *Auxiliary Lemmas:* The analysis is founded in Lemma 7, where the symmetric of the expected improvement is said to attain a positive infimum on the set $\hat{\mathcal{Z}}_\epsilon^c$. Lemma 6 will be instrumental in the proof of Lemma 7 but contains also some useful properties of function F and the solution set \mathcal{Z}^\star .

Lemma 6 (Basic properties). *Let F as defined in (15). Then the following properties hold.*

- 1) F is coercive;
- 2) $F^\star \geq 0$ and $\mathcal{Z}^\star \neq \emptyset$;
- 3) \mathcal{Z}^\star is compact;
- 4) If z^\bullet is block optimal for F in \mathcal{Z} , then it is global optimal for F in \mathcal{Z} .

Proof:

- 1) By Assumption 2 there is a path from each node i to some node j which is connected to an anchor k . Also, we know that, by definition, $\|y_{ij}\|$ and $\|w_{ik}\|$ are bounded by the ranges $d_{ij} < \infty$ and $r_{ik} < \infty$. So these components of z

will have no effect in the limiting behavior of F . If $\|x_i\| \rightarrow \infty$ there are two cases: (1) there is at least one edge $t \sim u$ along the path from i to j where $\|x_t\| \rightarrow \infty$ and $\|x_u\| \not\rightarrow \infty$, and so $h_{R_{tu}}(\|x_t - x_u - y_{tu}\|) \rightarrow \infty$; (2) if $\|x_u\| \rightarrow \infty$ for all u in the path between i and j , in particular we have $\|x_j\| \rightarrow \infty$ and so $h_{R_{ajk}}(\|x_j - a_k - w_{jk}\|) \rightarrow \infty$, and in both cases $F \rightarrow \infty$, thus, F is coercive.

- 2) Function F defined in (15) is a continuous, convex and real valued function lower bounded by zero; so, the infimum F^* exists and is non-negative. To prove this infimum is attained and $\mathcal{Z}^* \neq \emptyset$, we observe that the set \mathcal{Z} is a cartesian product of the closed sets \mathbb{R}^{np} , \mathcal{Y} and \mathcal{W} , and so \mathcal{Z} is also closed. Now consider the set $T_\alpha = \{z : F(z) \leq \alpha\}$; T_α is a sublevel set of a continuous, coercive function and, thus, it is compact. For some α , the intersection of T_α and \mathcal{Z} is nonempty and it is known that the intersection of a closed and a compact set is compact [30, Corollary to 2.35], so $T_\alpha \cap \mathcal{Z}$ is compact. As function F is convex, it is also continuous on the compact set $T_\alpha \cap \mathcal{Z}$, and by the extreme value theorem, the value $p = \inf_{z \in T_\alpha \cap \mathcal{Z}} F(z)$ is attained and it is obvious that $\inf_{z \in T_\alpha \cap \mathcal{Z}} F(z) = \inf_{z \in \mathcal{Z}} F(z)$.
- 3) $\mathcal{Z}^* = T_\alpha \cap \mathcal{Z}$ for $\alpha = F^*$, and we deduced in the previous proof that $T_\alpha \cap \mathcal{Z}$ is compact.
- 4) If z^\bullet is block-optimal, then $\langle \nabla F_i(z_i^\bullet), z_i - z_i^\bullet \rangle \geq 0$ for all $z_i \in \mathcal{Z}_i$ and for all i . When stacking the inequalities for all i , we get $\langle \nabla F(z^\bullet), z - z^\bullet \rangle \geq 0$, which proves the claim. ■

Lemma 7. Let ϕ be defined as (40), taking values on the set $\hat{\mathcal{Z}}_\epsilon^c$ in (36). Then,

1) Function ϕ is positive:

$$\phi(z) > 0, \quad \text{for all } z \in \hat{\mathcal{Z}}_\epsilon^c; \quad (41)$$

2) And, as a consequence, function ϕ is bounded by a finite positive value a_ϵ :

$$\inf_{z \in \hat{\mathcal{Z}}_\epsilon^c} \phi(z) = a_\epsilon. \quad (42)$$

Proof: We start by proving the first claim, $\phi(z) > 0$ for all $z \in \hat{\mathcal{Z}}_\epsilon^c$. Suppose $\phi(z) = 0$; then, by Equation (39)

$$F^i(z) = F(z),$$

which means z is block optimal; by Lemma 6, z is, then, global optimal, which contradicts the fact that z belongs to the set $\hat{\mathcal{Z}}_\epsilon^c$. The second claim follows by observing that ϕ is a sum of real valued functions and, thus, a real valued function, and that $\phi(z)$ is bounded below by zero in $\hat{\mathcal{Z}}_\epsilon^c$ and so it has a positive infimum for $\hat{\mathcal{Z}}_\epsilon^c$. ■

2) *Theorems:* Equipped with the previous Lemmas, we are now ready to prove the Theorems stated in Section V-B.

Proof of Theorem 3: We denote the random variable corresponding to the outcome of the t -th loop step of Algorithm 2 as Z^t . The expected value of the expected improvement function ψ is

$$\begin{aligned} \mathbb{E}[\psi(Z^t)] &= \mathbb{E}[\mathbb{E}[F(Z^{t+1}) | Z^t]] - \mathbb{E}[F(Z^t)] \\ &= \mathbb{E}[F(Z^{t+1})] - \mathbb{E}[F(Z^t)], \end{aligned}$$

where the second equality comes from the tower property documented, e.g., in Williams [31]. This expectation can also be written as

$$\begin{aligned} \mathbb{E}[\psi(Z^t)] &= \mathbb{E}[\psi(Z^t) | Z^t \in \hat{\mathcal{Z}}_\epsilon^c] \mathbb{P}(Z^t \in \hat{\mathcal{Z}}_\epsilon^c) \\ &\quad + \mathbb{E}[\psi(Z^t) | Z^t \notin \hat{\mathcal{Z}}_\epsilon^c] \mathbb{P}(Z^t \notin \hat{\mathcal{Z}}_\epsilon^c) \\ &\leq \mathbb{E}[\psi(Z^t) | Z^t \in \hat{\mathcal{Z}}_\epsilon^c] \mathbb{P}(Z^t \in \hat{\mathcal{Z}}_\epsilon^c). \end{aligned}$$

By combining both we get

$$\begin{aligned} \mathbb{E}[F(Z^{t+1})] - \mathbb{E}[F(Z^t)] \\ \leq \mathbb{E}[\psi(Z^t) | Z^t \in \hat{\mathcal{Z}}_\epsilon^c] \mathbb{P}(Z^t \in \hat{\mathcal{Z}}_\epsilon^c) \end{aligned}$$

which can be further bounded using Lemma 7 as

$$\mathbb{E}[F(Z^{t+1})] - \mathbb{E}[F(Z^t)] \leq -a_\epsilon p_t$$

where $p_t = \mathbb{P}(Z^t \in \hat{\mathcal{Z}}_\epsilon^c)$. By expanding the recursion we obtain

$$\mathbb{E}[F(Z^{t+1})] \leq -a_\epsilon \sum_{k=1}^t p_k + F(z^0),$$

which provides a bound on the sum of probabilities p_k when rearranged as

$$\sum_{k=1}^t p_k \leq \frac{F(z^0) - \mathbb{E}[F(Z^{t+1})]}{a_\epsilon}.$$

Taking t up to infinity, we obtain

$$\begin{aligned} \sum_{k=1}^{\infty} p_k &\leq \frac{F(z^0) - \mathbb{E}[F(Z(\infty))]}{a_\epsilon} \\ &\leq \frac{F(z^0) - \hat{f}^*}{a_\epsilon}. \end{aligned}$$

This means the infinite series of probabilities p_k assumes a finite value; by the Borel-Cantelli Lemma, we get

$$\mathbb{P}(Z^t \in \hat{\mathcal{Z}}_\epsilon^c, \text{ i.o.}) = 0,$$

where *i.o.* stands for *infinitely often*. This concludes the proof, since this statement is equivalent to the first claim of Theorem 3. ■

Proof of Theorem 4: Consider redefining the sets in (33) and (34) as

$$\begin{aligned} \mathcal{Y}_\epsilon &= \{z : F(z) < F^* + \epsilon\} \\ \mathcal{Y}_\epsilon^c &= \{z : F(z) \geq F^* + \epsilon\}, \end{aligned}$$

thus leading to

$$\hat{\mathcal{Y}}_\epsilon^c = \mathcal{Y}_\epsilon^c \cap \mathcal{Z}_{\hat{f}}.$$

Using the same arguments as in Lemma 7, we can prove that

$$\inf_{z \in \hat{\mathcal{Y}}_\epsilon^c} \phi(z) = b_\epsilon < \infty.$$

We now define a sequence of points \tilde{z}^t such that

$$\tilde{z}^t = \begin{cases} z^t & \text{if } z^t \in \hat{\mathcal{Y}}_\epsilon^c \\ z^* & \text{otherwise} \end{cases},$$

and the sequence of real values

$$\psi(\tilde{z}^t) = \begin{cases} \psi(z^t) & \text{if } z^t \in \hat{\mathcal{Y}}_\epsilon^c \\ 0 & \text{otherwise} \end{cases}.$$

The expected value of $\psi(\tilde{z}^t)$ is

$$\mathbb{E}[\psi(\tilde{z}^t)] = \mathbb{E}[F(\tilde{z}^{t+1})] - \mathbb{E}[F(\tilde{z}^t)].$$

Summing these expectations over time, we get

$$\sum_{k=0}^{t-1} \mathbb{E}[\psi(\tilde{z}^k)] = \mathbb{E}[F(\tilde{z}(t))] - F(z^0).$$

Taking t to infinity and interchanging integration and summation we obtain

$$\mathbb{E}\left[\sum_{k=0}^{\infty} \psi(\tilde{z}^k)\right] = \mathbb{E}[F(Z(\infty))] - F(z^0).$$

From the definition of $\psi(\tilde{z}^t)$ we can write

$$\mathbb{E}\left[\sum_{k=0}^{\infty} \psi(\tilde{z}^k)\right] \leq \mathbb{E}[K_\epsilon(-b_\epsilon)],$$

thus obtaining the result

$$\begin{aligned} \mathbb{E}[K_\epsilon] &\leq \frac{F(z^0) - \mathbb{E}[F(Z(\infty))]}{b_\epsilon} \\ &\leq \frac{F(z^0) - F^*}{b_\epsilon} \end{aligned}$$

which is a finite number. This completes the proof. ■

REFERENCES

- [1] P. Oğuz-Ekim, J. Gomes, J. Xavier, and P. Oliveira, "Robust localization of nodes and time-recursive tracking in sensor networks using noisy range measurements," *Signal Processing, IEEE Transactions on*, vol. 59, no. 8, pp. 3930–3942, Aug. 2011.
- [2] P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, and T.-C. Wang, "Semidefinite programming approaches for sensor network localization with noisy distance measurements," *Automation Science and Engineering, IEEE Transactions on*, vol. 3, no. 4, pp. 360–371, Oct. 2006.
- [3] Y. Shang, W. Rumi, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 15, no. 11, pp. 961–974, Nov. 2004.
- [4] G. Destino and G. Abreu, "On the maximum likelihood approach for source and network localization," *Signal Processing, IEEE Transactions on*, vol. 59, no. 10, pp. 4954–4970, Oct. 2011.
- [5] C. Soares, J. Xavier, and J. Gomes, "Simple and fast convex relaxation method for cooperative localization in sensor networks using range measurements," *Signal Processing, IEEE Transactions on*, vol. 63, no. 17, pp. 4532–4543, Sept. 2015.
- [6] J. Costa, N. Patwari, and A. Hero III, "Distributed weighted-multidimensional scaling for node localization in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 1, pp. 39–64, 2006.
- [7] G. Calafiore, L. Carlone, and M. Wei, "Distributed optimization techniques for range localization in networked systems," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, Dec. 2010, pp. 2221–2226.
- [8] C. Soares, J. Xavier, and J. Gomes, "Distributed, simple and stable network localization," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, Dec. 2014, pp. 764–768.
- [9] A. Ihler, I. Fisher, J. W., R. Moses, and A. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 4, pp. 809–819, Apr. 2005.
- [10] J. Ash and R. Moses, "Outlier compensation in sensor network self-localization via the EM algorithm," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, vol. 4, March 2005, pp. iv/749–iv/752 Vol. 4.
- [11] F. Yin, A. Zoubir, C. Fritsche, and F. Gustafsson, "Robust cooperative sensor network localization via the EM criterion in LOS/NLOS environments," in *Signal Processing Advances in Wireless Communications (SPAWC), 2013 IEEE 14th Workshop on*, June 2013, pp. 505–509.
- [12] P. Forero and G. Giannakis, "Sparsity-exploiting robust multidimensional scaling," *Signal Processing, IEEE Transactions on*, vol. 60, no. 8, pp. 4118–4134, Aug. 2012.
- [13] S. Korkmaz and A.-J. van der Veen, "Robust localization in sensor networks with iterative majorization techniques," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, Apr. 2009, pp. 2049–2052.
- [14] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.
- [15] S. Yousefi, X. W. Chang, and B. Champagne, "Distributed cooperative localization in wireless sensor networks without NLOS identification," in *Positioning, Navigation and Communication (WPNC), 2014 11th Workshop on*, March 2014, pp. 1–6.
- [16] D. Blatt and A. Hero, "Energy-based sensor network source localization via projection onto convex sets," *Signal Processing, IEEE Transactions on*, vol. 54, no. 9, pp. 3614–3619, Sept. 2006.
- [17] A. Simonetto and G. Leus, "Distributed maximum likelihood sensor network localization," *Signal Processing, IEEE Transactions on*, vol. 62, no. 6, pp. 1424–1437, Mar. 2014.
- [18] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms*. Springer-Verlag Limited, 1993.
- [19] M. Udell and S. Boyd, "Bounding duality gap for problems with separable objective," *Computational Optimization and Applications*, vol. 64, no. 2, pp. 355–378, 2016.
- [20] R. Phelps, "Convex sets and nearest points," *Proceedings of the American Mathematical Society*, vol. 8, no. 4, pp. 790–797, 1957.
- [21] R. B. Bapat, *Graphs and matrices*. Springer, 2010.
- [22] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.
- [23] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," in *Soviet Mathematics Doklady*, vol. 27, no. 2, 1983, pp. 372–376.
- [24] —, *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004.
- [25] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.
- [26] D. Shah, *Gossip algorithms*. Now Publishers Inc, 2009.
- [27] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21," <http://cvxr.com/cvx>, Apr. 2011.
- [28] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.
- [29] D. Jakovetic, J. Xavier, and J. Moura, "Cooperative convex optimization in networked systems: Augmented lagrangian algorithms with directed gossip communication," *Signal Processing, IEEE Transactions on*, vol. 59, no. 8, pp. 3889–3902, Aug. 2011.
- [30] W. Rudin, *Principles of Mathematical Analysis*, ser. International series in pure and applied mathematics. McGraw-Hill, 1976.
- [31] D. Williams, *Probability with martingales*. Cambridge university press, 1991.